



7. 影像壓縮

7.1 基本原理

7.2 評估準則

7.3 影像壓縮流程

7.4 資訊理論基礎

7.5 編碼技術

7.6 有損壓縮技術

7.7 JPEG影像壓縮標準



❁ 影像壓縮 (image compression) 的目的是要用較少的資料來表示同一張影像的內容，以減少電腦儲存空間、減少影像傳輸時間。
(不改變影像空間解析度、灰階解析度)

❁ 為什麼可以用較少的資料來表示同一張影像的內容呢？

可以減少多少資料量？

有沒有極限？

怎麼減少資料量？

影像品質是否會改變？

.. 等等，這些問題都是本章要探討的主題。

7.1 基本原理

✿ 影像資料能夠壓縮當然表示影像中有多餘的資料。找出多餘的資料，以達到影像資料壓縮的目的，其主要的技術來自於下列三方面的觀念：

- (i) 利用像素間關係 (interpixel correlation) 的描述或估計就可以減少資料量；例如，*DCT*、*DWT*、*DPCM*。
- (ii) 利用生理視覺的 (psychovisual) 特性可以創造出減少資料量的方法；例如，各種量化法 (quantization)、人眼對於高頻資訊不敏感的特性等。

- (iii) 二進位的編碼技術 (coding) 可以減少資料的使用量。以編碼技術創造出減少資料的方法；例如，*variable-length code*、*Huffman code*、*arithmetic code*。

✿ 一般影像壓縮技術就是整合這三類方法所創作出來的。所以上述三種觀念所創造出來的各種資料縮減方法就構成了一般影像壓縮技術的基礎。目前最重要的研究都集中在第一類技術上。



- ✿ 有些資料縮減方法將資料量變少，也同時造成影像品質的衰退。所以在允許影像品質衰退的方法中，我們要儘量減少影像品質的衰退，提高影像資料的縮小率。
- ✿ 影像壓縮的技術分為兩大類：一類是不會損害影像品質的無損壓縮技術 (lossless or error-free compression)，一類是較高壓縮率但會損害影像品質的有損壓縮技術 (lossy compression)；要選用那一種技術就完全看個人需要了。



7.2 評估準則

- ✿ 在影像壓縮的評估上，除了要考量資料的縮減量外，還要比較影像的品質。影像壓縮率 (compression ratio) 及 多餘資料 (redundancy) 有一定的定義。
- ✿ 假設原始影像資料量為 n_1 ，壓縮後的資料量為 n_2 ；影像壓縮率 C_R 則定義為
$$C_R = \frac{n_1}{n_2}$$
相對多餘資料 (relative data redundancy) R_D 則定義為
$$R_D = 1 - \frac{1}{C_R} = \frac{n_1 - n_2}{n_1}$$



❁ 壓縮率愈高，表示相對多餘資料愈多，去掉的資料愈多，保留的資料愈少。

❁ 影像品質的比較有多種評估準則。假設 $f(x, y)$ 是原始影像像素 (x, y) 的灰階， $g(x, y)$ 是壓縮解壓縮後同一像素的灰階，像素灰階的差異值為

$$e(x, y) = g(x, y) - f(x, y)。$$

我們可以從整張影像的差異值定義出各種品質的評估準則 (criterion)：



A. 平方誤差 (mean-square (*ms*) error)

$$e^2 = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^2(x, y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [g(x, y) - f(x, y)]^2。$$

B. 平方根誤差 (root-mean-square (*rms*) error)

$$e_{rms} = \sqrt{e_{ms}}。$$

C. 信號雜訊比或訊噪比

(mean-square signal-to-noise ratio (*SNR*))

$$SNR_{ms} = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g^2(x, y)}{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^2(x, y)}。$$

D. 平方根訊噪比 (*rms* value of SNR)

$$SNR_{rms} = \sqrt{SNR_{ms}}.$$

E. 峰值訊噪比

(peak value signal-to-noise ratio (*PSNR*))

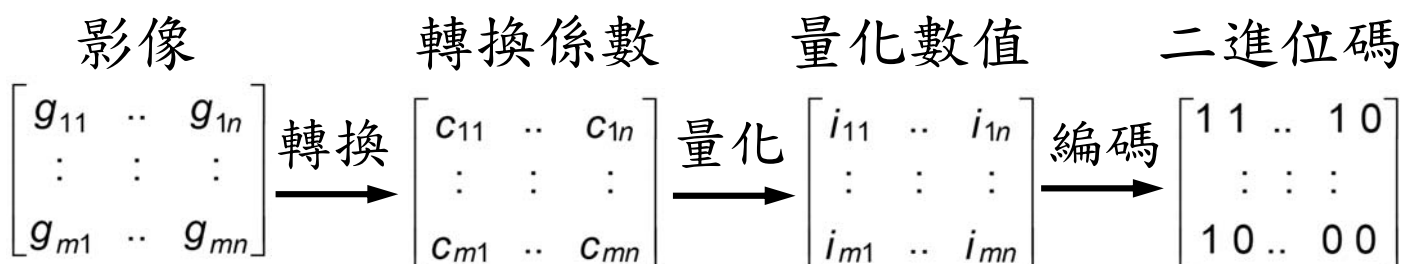
$$PSNR = \frac{[\text{peak value of } g(x,y)]^2}{e_{ms}}.$$

目前大多數影像或視訊品質的評估準則，都是從上式修改來的另一個峰值訊噪比公式

$$PSNR = 10\log_{10} \frac{255^2}{e_{ms}} \text{ (dB) 分貝 (decibel)}$$

7.3 影像壓縮流程

- ❁ 前述一般影像壓縮技術是整合分析像素間的關係、生理視覺的特性、及編碼技術而成的。現在我們就來看看一般影像壓縮的流程，



上述轉換、量化、編碼個別對應分析像素間的關係、生理視覺的特性、及編碼技術。

7.3.1 轉換

❁ 轉換 (transformation) 是分析像素間之關係的通式。通常轉換本身不具有減少資料量的功能，但它一定是要協助後面兩個步驟的工作更有效率。

❁ 線性轉換 (linear transformation) 是最簡單的一種轉換。線性轉換的通式為

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{或} \quad \mathbf{y} = \mathbf{A} \mathbf{x}。$$

❁ 有些線性轉換是可逆 (reversible) 有些是不可逆；端視 \mathbf{A} 矩陣而定。一般可逆轉換一定不會損害資料，且可直接做反轉換，所以較常被使用。

❁ 線性轉換範例

一、傅立葉轉換、離散餘弦轉換、小波轉換

二、列長碼 (run length encoding)

二值化影像中的一列，

001111000000001111111011

表示成列長碼為 0 2 4 8 7 1 2；

格式為 $g_1 \ l_1 \ l_2 \ \cdots \ l_k$ 。



三、差分碼 (difference code)


$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix}$$

若原始資料為 $[y_1 \ y_2 \ y_3 \ \dots \ y_n]^T$ ，則轉換後的資料變成 $[y_1 \ y_2 - y_1 \ y_3 - y_2 \ \dots \ y_n - y_{n-1}]^T$ 。




7.3.2 量化

- ✿ 利用生物視覺的特性縮減資料量。例如，一張彩色影像，每一像素用 24 位元表示。如果把像素縮減成只用 16 位元，眼睛看起來好像差不多，當然就可以縮減影像資料量為原來的三分之二。
- ✿ 若將像素縮減成只用 8 位元，眼睛看起來有些差別，我們還是可以用特殊的量化方法“改良灰階尺度量化”(improved gray-scale (IGS) quantization)，修改影像成看不出差異，這時就可以縮減影像資料量為原來的三分之一。

- 

Improved gray-scale (IGS) quantization

改良式灰階量化法的結果僅適合眼睛觀看或做影像壓縮。該方法就影像一列列掃描，再利用類似半色調列印的誤差擴散觀念來創造沒有假邊的量化影像
- 

步驟 (8 位元 → 4 位元)

S1. 首先將影像灰階表示成 8 位元的二進位字串；影像每一列獨自處理。

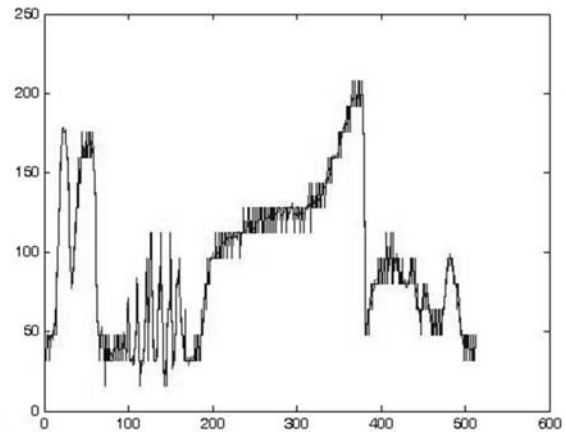
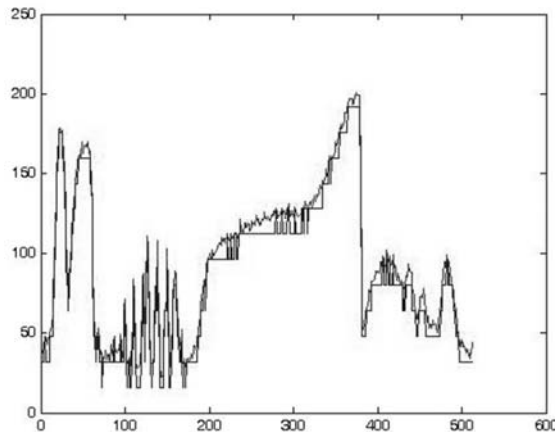
S2. 令 4 元字串 $sum = 0$ (“0000”)。

S3. 每一列像素從左至右處理，將目前像素的二元字串加 sum ，將相加結果的最左邊 (最重要) 4 位元輸出成量化結果，最右邊 (不重要) 4 位元設為 sum 。重複下一個像素，直到一列所有像素都處理完，再換下一列。當某一像素的最左邊 4 位元已經是 “1111” 時，則該像素不加 sum

以下表的 256 灰階量化成 16 灰階為例。

灰階	二元字串	sum	輸出灰階
48	0011 0000	0011 0000	48 (3)
50	0011 0010	0011 0010	48 (3)
52	0011 0100	0011 0110	48 (3)
54	0011 0110	0011 1100	48 (3)
56	0011 1000	0100 0100	64 (4)
58	0011 1010	0011 1110	48 (3)
60	0011 1100	0100 1010	64 (4)
62	0011 1110	0100 1000	64 (4)
64	0100 0000	0100 1000	64 (4)
252	1111 1100	1111 1100	240 (15)

The last row



均勻量化結果。

改良式灰階量化結果。

7.3.3 編碼

- ❁ 編碼 (coding) 是給予各種符號或灰階一個二進位碼。
- ❁ 編碼一定要能唯一解碼 (uniquely decodable)，所以編碼一定不會損害影像品質。
- ❁ 編碼的方式有多種分類；例如，
等長碼 (fixed-length or equal-length code) 與
變長碼 (unequal-length or variable-length code)
即時碼 (instantaneous code) 與非即時碼、
熵碼 (entropy code) 與非熵碼。

7.4 資訊理論基礎

- ❁ 本節的資訊理論 (information theory) 只是探討在已知資料的分佈 (distribution) 情形下，一堆資料或一張影像的平均二進位碼長的下限 (low bound) 是多少。
- ❁ 熵 (entropy) 是度量一堆隨機亂數數值亂度 (randomness) 的準則。假設有 m 個隨機亂數值 $\alpha_1, \alpha_2, \dots, \alpha_m$ ； α_i 的機率值為 $p(\alpha_i)$ ，簡寫為 p_i 。這一堆隨機亂數值的熵定義為

$$H = - \sum_{k=1}^m p_k \log_2 p_k$$

其中的機率值 p_k 都小於等於 1，對數 (log) 值都小於等於 0；所以取負號，讓熵值大於 0。接下來看兩個極端的例子，

$$m = 8, p_1 = p_2 = \dots = \frac{1}{8} \rightarrow H = - \sum_{k=1}^8 \frac{1}{8} \log_2 \frac{1}{8} = 3$$

資料最亂，每一種數值 (符號) 都有，而且每種符號出現的機率都一樣；所以熵最大。

$$m = 8, p_1 = 1, p_2 = 0, \dots, p_8 = 0 \Rightarrow H = - \sum_{k=1}^8 p_i \log_2 p_i = 0$$

$$H = - (1 * 0 + 0 * ? + 0 * ? + \dots + 0 * ?)$$

資料最一致的，只有一種符號 α_1 出現；所以熵最小。



- ✿ 在編碼應用上，熵是用來表示一堆資料的平均二進位碼長之下限 (low bound)。例如，影像有 256 種灰階 $\{g_i\}$ ，各灰階出現的機率值為 $\{p_i\}$ ；則編碼後的平均二進位碼長不可能小於

$$H = - \sum_{k=0}^{255} p_k \log_2 p_k$$

位元。所以如果我們設計一組編碼 (code word) $\{c_0, c_1, \dots, c_m\}$ ，各碼長為 $\{\beta_0, \beta_1, \dots, \beta_m\}$ ，則該

影像平均每個像素需要 $R = \sum_{k=0}^m \beta_k p_k$ 位元。

如果 R 接近 H ，則表示這一組碼是近乎最佳碼，稱為熵碼 (entropy coder)。



7.5 編碼技術

- ✿ 在本節中我們將介紹幾種編碼 (coding) 技術，這些技術都一定可以唯一解碼 (uniquely decodable codes)，而且不會損害原始資料品質。
- ✿ 編碼的對象可能是影像像素的灰階或轉換後的係數值；但我們都稱為符號。

7.5.1 霍夫曼編碼

- 霍夫曼編碼 (Huffman code) 是一種緊湊碼，要使用到碼的機率值；它是少數具有平均最短碼 (minimum-length code) 的方法之一。
- 霍夫曼編碼的兩個主要步驟為：
 - S1. 掃描整張影像，統計符號種類，計算各種符號的機率值。將符號依機率值大小排序。重複結合兩個最小機率值的符號群成單一個符號群，再重新依機率值大小排序。
 - S2. 依照原來合群的反次序，依序指定各符號群不等長碼。

- 以一張 8 灰階影像為例。使用三位元的二進位等長碼，每一像素要使用 3 位元；如果使用霍夫曼編碼，則平均每一像素只需使用 2.62 位元。

r_k	$P_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.07	000	3	111	3
$r_1 = 1/7$	0.11	001	3	011	3
$r_2 = 2/7$	0.23	010	3	10	2
$r_3 = 3/7$	0.32	011	3	00	2
$r_4 = 4/7$	0.04	100	3	11000	5
$r_5 = 5/7$	0.16	101	3	010	3
$r_6 = 6/7$	0.02	110	3	11001	5
$r_7 = 1$	0.05	111	3	1101	4

步驟 1. 機率結合程序

符號	機率	1	2	3	4	5	6
g_3	0.32	0.32	0.32	0.32	0.32	0.41	0.59
g_2	0.23	0.23	0.23	0.23	0.27	0.32	0.41
g_5	0.16	0.16	0.16	0.18	0.23	0.27	
g_1	0.11	0.11	0.11	0.16	0.18		
g_0	0.07	0.07	0.11	0.11			
g_7	0.05	0.06	0.07				
g_4	0.04	0.05					
g_6	0.02						










K 種碼，要做 $K-2$ 次合併

步驟 2. 指定二進位碼

符號	機率	1	2	3	4	5	6
g_3	0.32 00	0.32 00	0.32 00	0.32 00	0.32 00	0.41 1	0.59 0
g_2	0.23 10	0.23 10	0.23 10	0.23 10	0.27 01	0.32 00	0.41 1
g_5	0.16 010	0.16 010	0.16 010	0.18 11	0.23 10	0.27 01	
g_1	0.11 011	0.11 011	0.11 011	0.16 010	0.18 11		
g_0	0.07 111	0.07 111	0.11 110	0.11 011			
g_7	0.05 1101	0.06 1100	0.07 111				
g_4	0.04 11000	0.05 1101					
g_6	0.02 11001						

K 種碼，最長碼長度為 $K-1$

❁ 上述霍夫曼編碼範例的熵值為 2.57，平均碼長為 2.62 位元。相差小於 2 %。所以霍夫曼編碼是屬於熵碼的一種。



7.5.2 B-碼

- ✿ B -碼 (B -codes) 有多種的編法； B -碼中的 c 位元稱為連續位元 (continuation bit)，它的數值可以是0或1；但代表同一符號之二進位碼的 c 位元要都相同。

自然碼	B_1 碼	B_2 碼
0	c0	c00
1	c1	c01
00	c0c0	c10
01	c0c1	c11
10	c1c0	c00c00
11	c1c1	c00c10
000	c0c0c0	c00c10
001	c0c0c1	c00c11
010	c0c1c0	c01c00



- ✿ 以 B_1 - 碼 0011010111000001 為例，一次檢查 2 碼 (B_2 - 碼，一次檢查 3 碼)，各碼為 00, 11, 0101, 11, 000001。
- ✿ 從上例的解碼過程中可以發現， B -碼不是即時碼，也就是要解出一個碼時，要多看一個 c 位元才能知道前面位元已構成一個碼了。
- ✿ B -碼不需要各碼出現的機率值， B -碼也不是熵碼，所以效能比霍夫曼編碼及算術編碼差。



7.5.3 位移碼

- ✿ 位移碼 (shift codes, S_n -code) 有多種的編法。
 S_n -碼是以 n 位元的自然二進位碼來表示。當使用超過 2^n-1 個碼時，就會讓 2^n 個碼中的最後一碼 “111 ..1” (有 n 個 1)，向左位移 n 個位元，後面再補 n 個自然二進位碼的位元；也就是使用了 $2n$ 個位元來表示其他的碼。
- ✿ S_n -碼是即時碼。 S_n -碼和 B -碼一樣不用各碼出現的機率值， S_n -碼也不是熵碼，所以效能比霍夫曼編碼及算術編碼差。
- ✿ 位移碼在輸入符號的出現機率值呈狹義單調遞減 (monotonically decreasing) 時，特別有效率。



S_2 - 碼	S_3 - 碼
00	000
01	001
10	010
1100	011
1101	100
1110	101
111100	110
111101	111000
111110	111001

7.6 有損壓縮技術

- ✿ 在影像壓縮應用上，有損壓縮 (lossy compression) 技術可以獲得較高的壓縮率；所以目前在理論或實務應用上都是主流。
- ✿ 有損壓縮 的技術非常多；例如，
 - KLT* (Karhunen-Loeve Transform)
 - DFT* (Discrete Fourier Transform)
 - DCT* (Discrete Cosine Transform)
 - VQ* (Vector Quantization)
 - Predictive coding

DM (Delta Modulation)
DPCM (Differential Pulse Code Modulation)
SB (Subband Encoding)
Model-based Image Compression
Fractal Image Compression

✿ *JPEG*

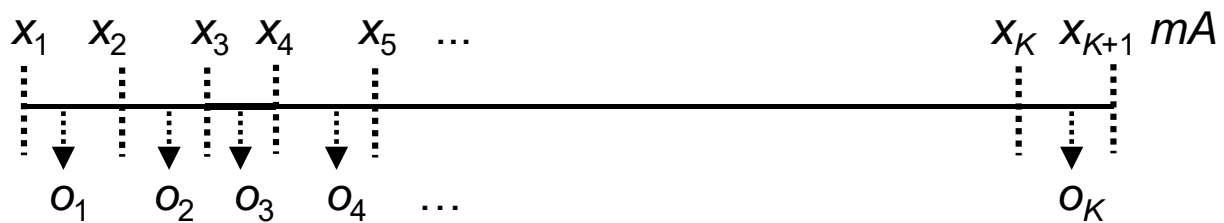
JPEG 是利用 *DCT* 做靜態影像壓縮的標準。它的理論是建立在人眼的特性上：

- (1) 人眼對彩度的敏感性比亮度差；
- (2) 人眼對高頻資訊的敏感性比低頻資訊差。



差分脈衝調變編碼

- ❁ 脈衝調變編碼 (pulse-code modulation, *PCM*) 是一種類比訊號轉數位訊號的方法。一個 *PCM* 串列中的訊號數值會量化在一個事先定義好的範圍內。一個 *PCM* 串列的擬真度 (fidelity) 會受到兩個參數影響；一是取樣率 (sampling rate), 每單位時間取幾個樣本；一是樣本尺度 (bit depth), 每一個樣本數據最多有幾種變化；例如, $2^8, 2^{10}, \dots$ 等。



- ❁ 差分脈衝調變編碼 (differential pulse code modulation, *DPCM*) 是在脈衝編碼調變串列中，記錄訊號相減數據 (稱為差分)，而不是直接記錄量化後的訊號數據。紀錄差分的好處是能夠集中將數據分佈 (distribution)；如此在使用不等長編碼法 (unequal-length coding) 時，可以減少資料量。
- ❁ 訊號相減的方式有兩種：一是相鄰訊號相減；例如，一串訊號 $s_1, s_2, s_3, \dots, s_n$ 改成為 $s_1, s_2 - s_1, s_3 - s_2, \dots, s_n - s_{n-1}$ 。另一個是以各別原始訊號與預測訊號相減。

❁ 假設原始訊號為 $s_1, s_2, s_3, \dots, s_n$ 。差分數值為

$$e_i = s_i - \hat{s}_i$$

其中是 s_i 的預測值。有多種預測的方法，其中以線性預測法最簡單。用 5×5 遮罩

定義線性預測模式

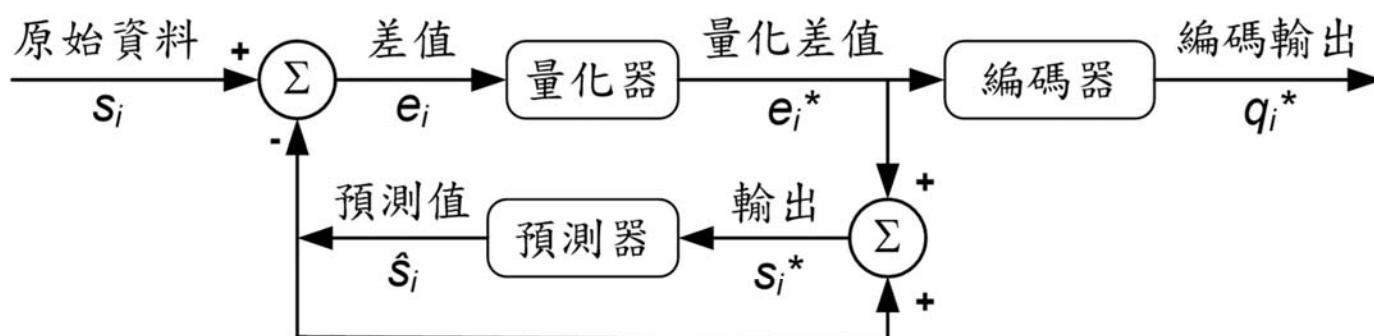
$$\hat{s}_i = \sum_{j=1}^{12} a_j s_j$$

s_1	s_2	s_3	s_4	s_5
s_6	s_7	s_8	s_9	s_{10}
s_{11}	s_{12}	s_i		

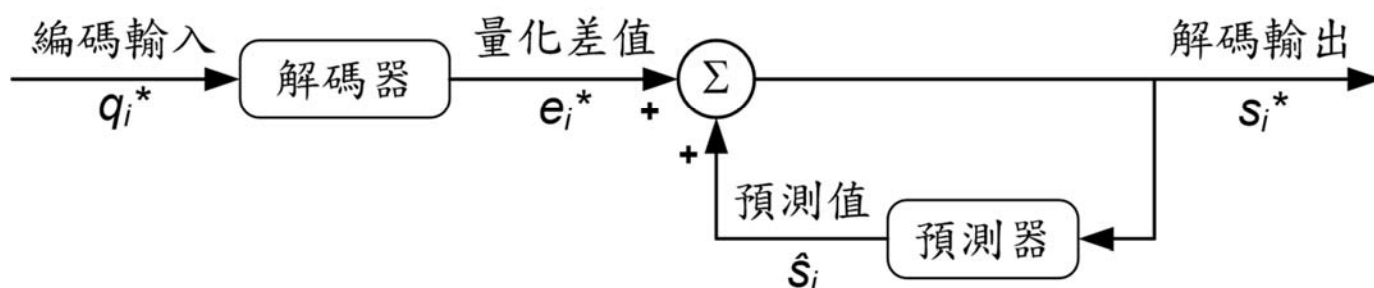
其中 $s_1 \sim s_{12}$ 是在列掃描次序下，排在 s_i 前的相鄰像素。這些參數可以用最小平方誤差準則來估計。最小平方誤差準則是在 ε 最小的情形下求出所有參

數 a_j ，其中 $\varepsilon = \sum_{i=1}^m (s_i - a_1 s_{1i} - a_2 s_{2i} - \dots - a_n s_{ni})^2$

❁ 差分脈衝調變編解碼 (DPCM) 流程



(a) 編碼流程。

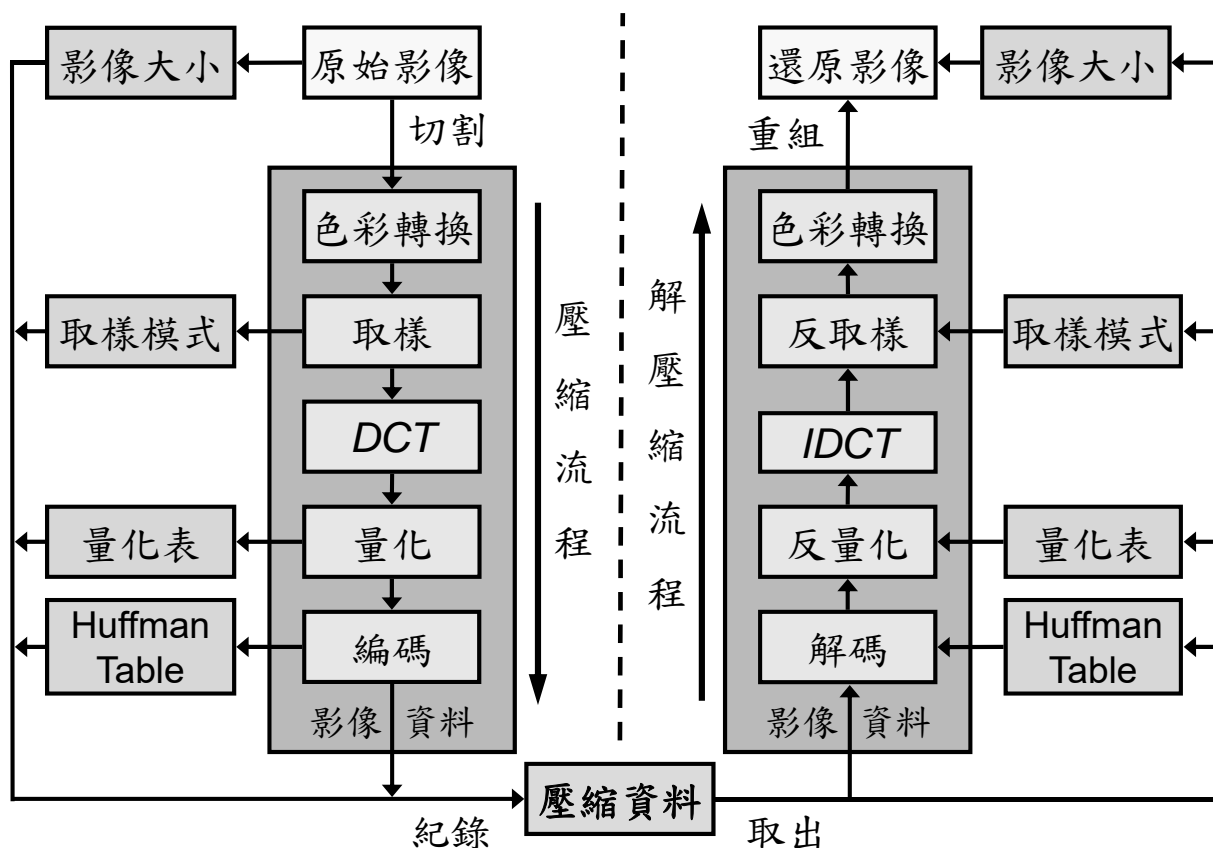


(b) 解碼流程。

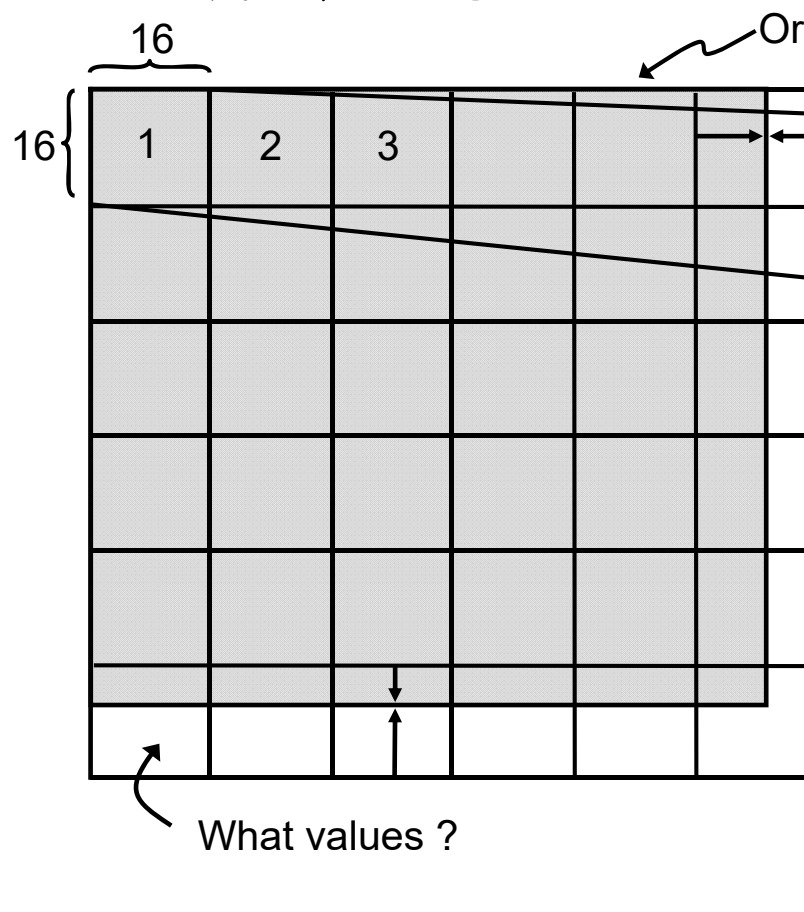
7.7 JPEG 壓縮與檔案格式

- ✿ *JPEG* 是利用 *DCT* 做靜態影像壓縮的標準。它的理論是建立在人眼對於高頻信號不敏感的特性之上。
- ✿ *JPEG* 有多種不同效率的壓縮方式：(i) 標準基本模式 (standard baseline mode)、(ii) 最佳基本模式 (optimized baseline mode)、及 (iii) 漸進傳輸模式 (progressive mode)。
- ✿ *JPEG* 是 ISO 國際標準協會 Joint Photographic Experts Group 所訂定的靜態影像壓縮標準。*JPEG* 的檔案格式稱為 *JPEG File Interchange Format (JFIF)*。

7.7.1 JPEG 壓縮/解壓縮的流程



A. 切割基本區塊



- 以16x16 pixels 為一個大區塊 (macro block)。
- 每一個大區塊再分割成 2x2 區塊 (blocks)。
- 色彩轉換及取樣處理是以大區塊為單位。
- *DCT* 及量化是以區塊為處理單位。
- 由左到右，由上到下的順序處理。

B. 色彩轉換 (Color transformation)

✿ $YC_bCr \Leftarrow RGB$

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

✿ $RGB \Leftarrow YC_bCr$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344136 & -0.714136 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

✿ 更有效率的 RGB 轉 YC_bCr 公式

$$Y = 0.299 (R - G) + G + 0.114 (B - G)$$

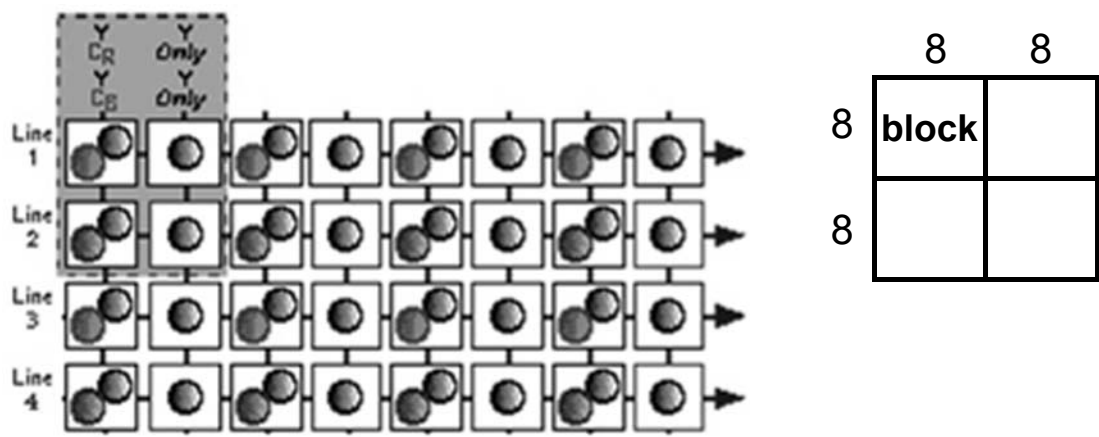
$$C_b = 0.564 (B - Y)$$

$$C_r = 0.713 (R - Y)$$

C. 取樣 (Sampling)

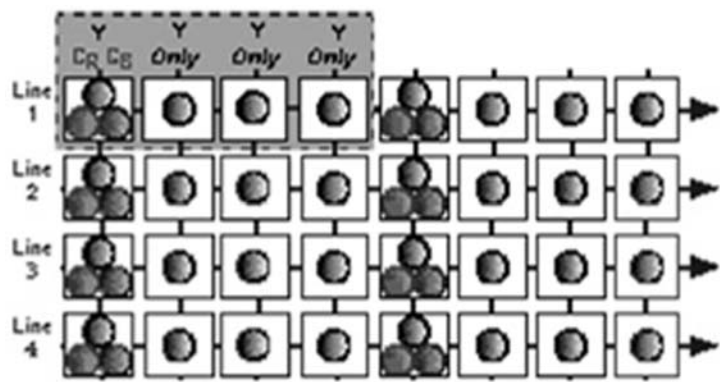
✿ *YCbCr* 有三種取樣模式:

(1) **4 : 2 : 0**



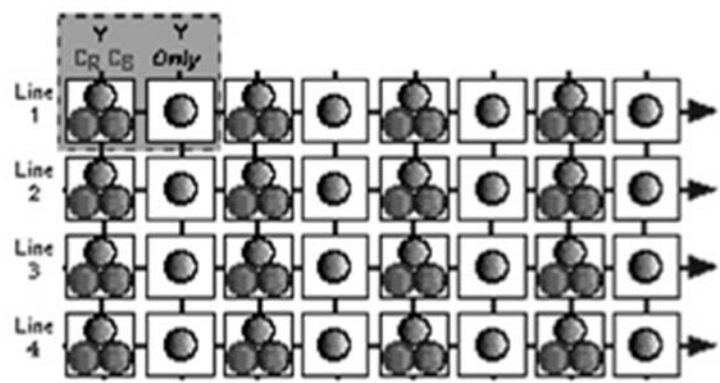
在 2×2 區塊中，有 4 個 *Y* 資料區塊，但各只有一個 *Cb* 及 *Cr* 資料區塊。總共 6 個資料區塊形成一個最小編碼單元 (minimum coded unit, *MCU*)。

(2) **4 : 1 : 1**



在 1×4 區塊中，有 4 個 *Y* 資料區塊，但各只有一個 *Cb* 及 *Cr* 資料區塊。總共 6 個資料區塊形成一個最小編碼單元。

(2) **4 : 2 : 2**



在 1×2 區塊中，有 2 個 *Y* 資料區塊，但各只有一個 *Cb* 及 *Cr* 資料區塊。總共 4 個資料區塊形成一個最小編碼單元。



D. DCT (Discrete Cosine Transform)

S1. 先將 Y block 中的每個 Y 值減去 128。

S2. 再為 block 做 2D-DCT 轉換：

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x,y) \left[\cos \frac{(2x+1)u\pi}{16} \right] \left[\cos \frac{(2y+1)v\pi}{16} \right],$$

$$\text{where } \alpha(i) = \begin{cases} \sqrt{1/8} & \text{for } i=0 \\ \sqrt{2/8} & \text{for } i=1 \sim 7 \end{cases}$$

✿ 為了計算效率，用兩次 1D-DCT 取代一次 2D-DCT:

$$d(x,v) = \sum_{y=0}^7 f(x,y) \alpha(v) \left[\cos \frac{(2y+1)v\pi}{16} \right]$$

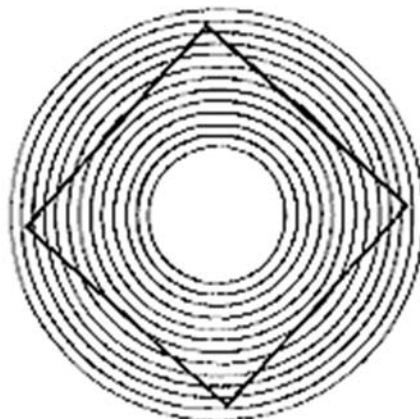
$$c(u,v) = \sum_{x=0}^7 d(x,v) \alpha(u) \left[\cos \frac{(2x+1)u\pi}{16} \right]$$



並且將 cos 函數與 α 函數合併做成 8x8 lookup table，未來使用時就不用再計算：

$$A[a(i,j)]_{8 \times 8},$$

$$\text{where } a(i,j) = \alpha(i) \cos \frac{(2j+1)i\pi}{16}.$$



E. 量化 (quantization)

- 量化的方式是將 block 中的數值個別除以量化表中的數值 $Q(i,j)$ ，再取四捨五入整數。

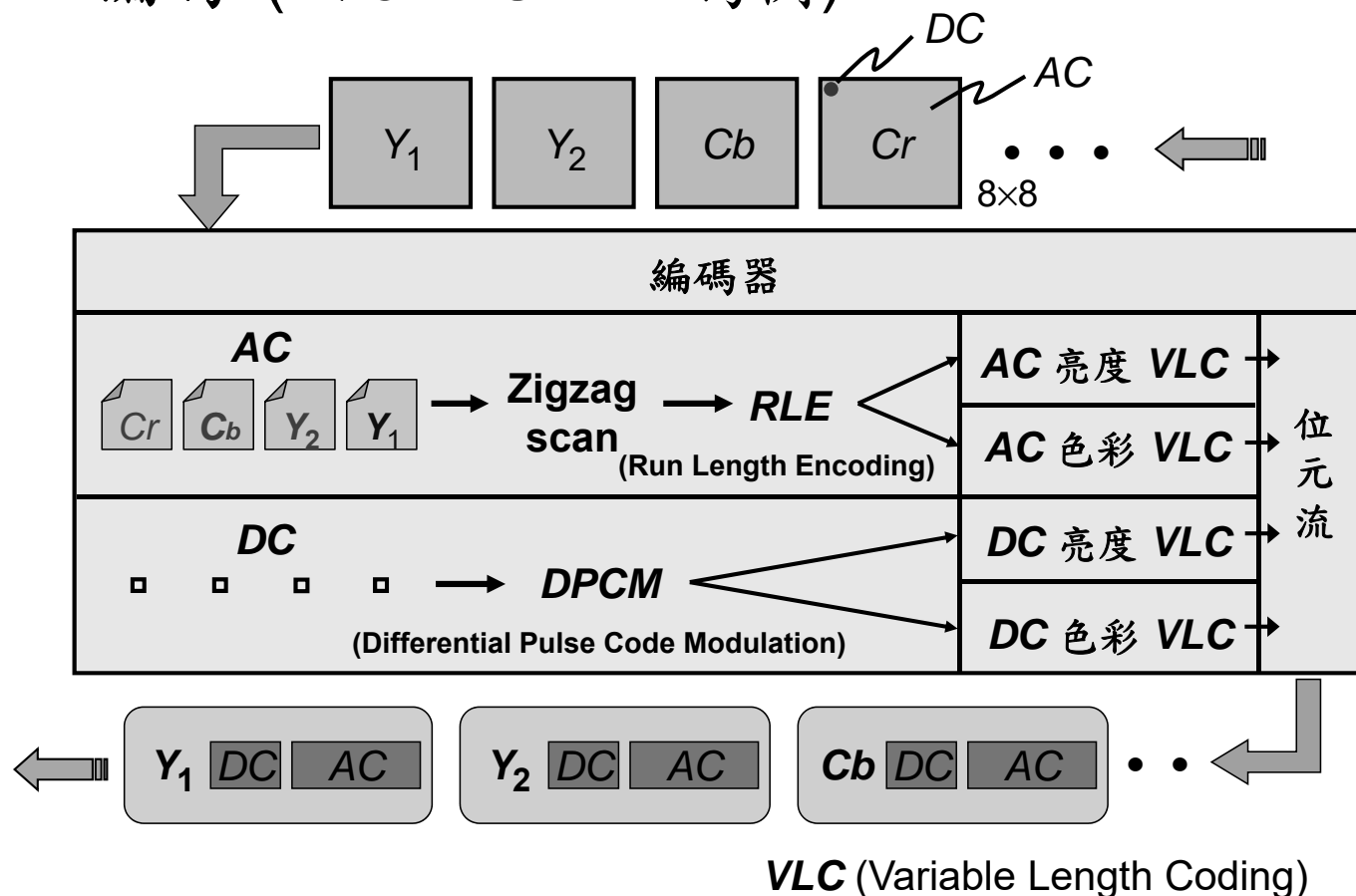
$$[block]_{8 \times 8} = [block]_{8 \times 8} \div [Quantization_table]_{8 \times 8}$$

- 量化表有預設的，也可以自行設計放在檔案中。
- 下表是兩個預設量化表例子

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

F. 編碼 (以 JPEG 422 為例)



DPCM (Differential Pulse Code Modulation)

✿ DPCM 是取每個 *block* 的 *DC* 值與前一個 *block* 的 *DC* 值之差質來做 *DC VLC* 編碼：

$$Diff\ DC(i) = DC(i) - DC(i-1)$$

✿ 前一個 *block* 的 *DC* 值在此稱為預測值，而第一個 *DC* 的預測值為 0。

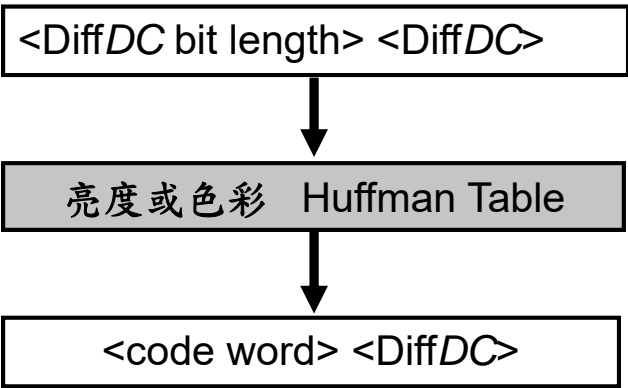
✿ 範例

DC 值： 1 3 5 7 9 7 5 3 1

預測值：- 0 1 3 5 7 9 7 5 3

DiffDC： 1 2 2 2 2 -2 -2 -2 -2

DC LVC (1/3)



DiffDC bit length	DiffDC
0	0
1	-1; 1
2	-3,-2; 2,3
3	-7..-4; 4..7
4	-15..-8; 8..15
5	-31..-16; 16..31
6	-63..-32; 32.. 63
7	-127..-64; 64..127
8	-255..-128; 128..255
9	-511..-256; 256..511
10	-1023..-512; 512..1023
11	-2047..-1024; 1024..2047

✿ Huffman Table (in next page) 中的 <code word> 用於編碼 <DiffDC bit length>。

DC LVC (2/3)

Huffman Table for brightness		
DiffDC bit length	code word length	code word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Huffman Table for chromaticity		
DiffDC bit length	code word length	code word
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

DC LVC (3/3)

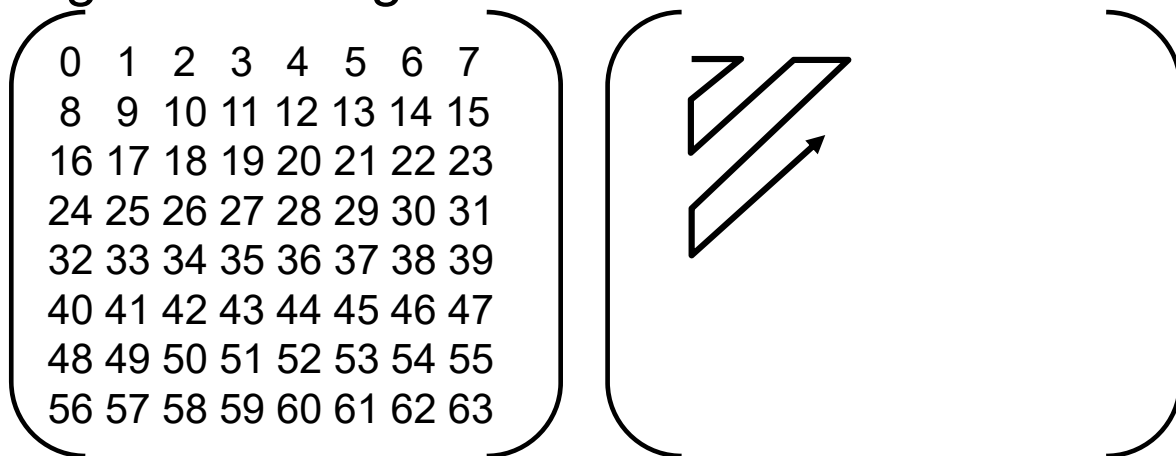
範例 (Brightness)

<length> <DiffDC> <length> <DifDC> = <4> <15> <2> <-3>
 ↓ Huffman Table
 <code> <DiffDC> <code> <DiffDC> = <101> <15> <011> <-3>
 ↓ binary DiffDC
 <code> <DiffDC> <code> <DiffDC> = <101> <1111> <011> <00>

- 正負號的表達以 2's complement 表示。
- DiffDC 的第一個 bit 如果為 ‘1’，則表示 DiffDC 為正數；反之，則為負數。取值時先做 complement 然後加上負號。
- 編碼時，要把 VLC 的 Huffman Table 也儲存下來。

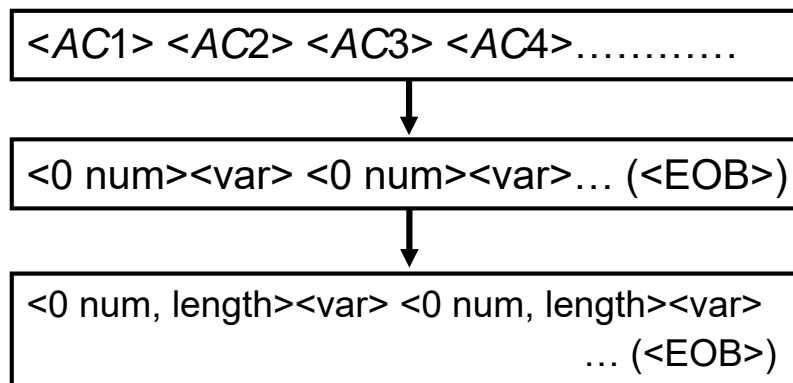
Zigzag Scan

- Zigzag 掃描是把陣列值由左上到右下依 Z 字排列；目的是將 2D 的資料轉為 1D，以利後續的 Run Length Encoding。



0, 1, 8, 16, 9, 2, 3, 10, 17, 24, 32, 25, 18, 11, 4, 5, 12, 19, 26, 33, 40,
 48, 41, 34, 27, 20, 13, 6, 7, 14, 21, 28, 35, 42, 49, 56, 57, 50, 43, 36,
 29, 22, 15, 23, 30, 37, 44, 51, 58, 59, 52, 45, 38, 31, 39, 46, 53, 60,
 61, 54, 47, 55, 62, 63

RLE (Run Length Encoding) (1/2)



length (bits)	AC value
0	0
1	-1;1
2	-3,-2; 2,3
3	-7..-4; 4..7
4	-15..-8; 8..15
5	-31..-16; 16..31
6	-63..-32; 32.. 63
7	-127..-64; 64..127
8	-255..-128; 128..255
9	-511..-256; 256..511
10	-1023..-512; 512..1023



RLE (Run Length Encoding) (2/2)

❁ 範例

<ACs.....> = 0, -1, -2, 0, 0, 0, 1, 0, 0, ... 0

<0 num> <var> <EOB> = <1> <-1> <0> <-2> <3> <1> ... <EOB>

↓ var length

<0 num, length> <var> ... <EOB> = <1, 1> <-1> <0, 2> <-2> <3, 1> <1> ..
<EOB>

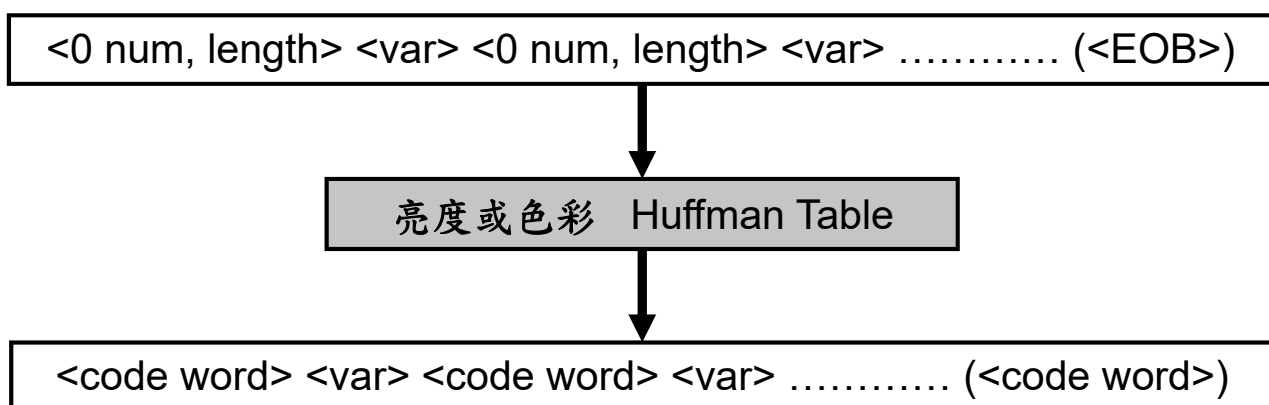
❁ RLE 中的符號 <0 num, length> 其大小為 1byte，0 num 和 length 各佔 4 bits。如果 0 的連續個數超過 16 個且沒有持續到尾端時，<0 num> <var> 編碼將變成 <16><0>，一共表示 17 個 0。

❁ 編碼中的 <EOB> 符號，以實際數值來表示為 <0, 0>。

❁ 如果編碼尾端不是以 0 結尾 (也就是說 block 的最後一個值不是 0) 的話，則編碼不用加上 <EOB>。



AC VLC



❁ Huffman Table 中 <code word> 用於編碼 <0 num, length>。

Huffman Table (1/2)

- ✿ 編碼端會傳遞 16 個 bytes，表示長度從 1~16 code word 的個數。
- ✿ 假設共有 n 個 code word，則會再傳遞 n 個 bytes，表示各 code word 所代表的 symbol 值。
- ✿ 解碼端重建 code word 的原則是：code word 初始化為 0。同樣長度的 code word 其值以加一累進。在增加 code word 長度時，要先把 code word 加一，然後再往左 shift 1 bit，也就是在尾端補上一個 0。

Huffman Table (2/2)

✿ 範例

假設傳輸了下列內容，則還原的 Huffman Table 如下表所示：

00 03 01 01 01 01 01 01 01 01 01 00 00 00 00 00

00 01 02 03 04 05 06 07 08 09 0A 0B

第一列為各長度 code word 的個數

第二行為符號值；

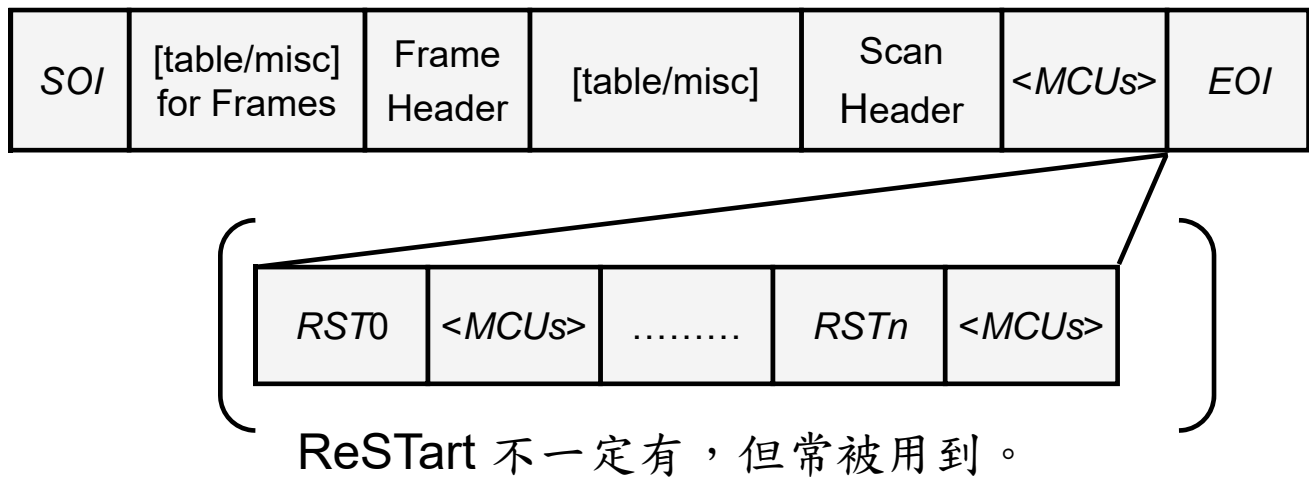
DC 的符號值是 <DiffDC bit length>

AC 的符號值 <0 num, length>。

DiffDC bit length	code word length	code word
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

7.7.2 JPEG 檔案格式

❁ *JPEG* (baseline mode) 檔案格式稱為 *JFIF* (*JPEG* File Interchange Format)



Decoding 必要的資訊

1. 影像的長寬 (in Frame Header)
2. 取樣模式 (in Frame Header)
3. 量化表、Huffman Table (in table/misc)
4. 使用表格編號等 (in Scan Header)
5. 影像資料 (*YCbCr*) (in *MCUs*)

JPEG 檔案範例 (1/2)

SOI

Table/misc

量化表

Frame Header

Table/misc

Huffman Table

```
00000000h: FF D8 FF EO 00 10 4A 46 49 46 00 01 01 01 00 64 ; ??JFIF.....d
00000010h: 08 07 08 0A 0A 09 0B 0E 18 0F 0E 0D 0D 0E 1D 15 ; .....
00000020h: 00 64 00 00 FF DB 00 43 00 09 06 07 08 07 06 09 ; .....
00000030h: 16 11 18 22 1E 24 24 22 1E 21 21 26 2B 37 2E 2A ; ..+7.&
00000040h: 28 34 29 21 21 30 41 30 34 39 3A 3D 3E 3D 25 2E ; (4)!!0A049:>=%.
00000050h: 43 48 43 3C 48 37 3C 3D 3B FF DB 00 43 01 0A 0A ; CHC<H7<=: ?C...
00000060h: 0A 0E 0C 0E 1C 0F 0F 1C 3B 27 21 27 3B 3B 3B 3B ; 
00000070h: 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B ; 
00000080h: 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B ; 
00000090h: 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B ; 
000000a0h: 00 11 08 00 0F 00 0F 03 01 22 00 02 11 01 03 11 ; :;;;;;;;;
000000b0h: 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 ; . ?.....
000000c0h: 00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 ; .....
000000d0h: 0A 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05 ; .. ??.....
000000e0h: 05 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21 ; .....}.....!
000000f0h: 31 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23 ; 1A..Qa."q.2??#
00000100h: 42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 ; B掄.R掄$3br?...
00000110h: 18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A ; ...%&'()*456789:
00000120h: 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A ; CDEFGHIJSTUVWXYZ
00000130h: 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A ; cdefghijstuvwxyz
00000140h: 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 ; ????????
00000150h: 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 ; 上它夾帶眾斯須
00000160h: B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 ; 號甄藥齏 均助煙
00000170h: D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 ; 第嶠答脛齏細鴉膳
00000180h: F2 F3 F4 F5 F6 F7 F8 F9 FA FF C4 00 1F 01 00 03 ; 觀鑊論闕??....
```

JPEG 檔案範例 (2/2)

Huffman Table

Table/misc

Scan Header

MCUs

EOI

```
00000190h: 01 01 01 01 01 01 01 01 01 01 00 00 00 00 00 01 ; .....
000001a0h: 02 03 04 05 06 07 08 09 0A 0B FF C4 00 B5 11 00 ; ..... ??
000001b0h: 02 01 02 04 04 03 04 07 05 04 04 00 01 02 77 00 ; .....w.
000001c0h: 01 02 03 11 04 05 21 31 06 12 41 51 07 61 71 13 ; .....!1..AQ.aq.
000001d0h: 22 32 81 08 14 42 91 A1 B1 C1 09 23 33 52 F0 15 ; "2?.B 掄.#3R?
000001e0h: 62 72 D1 0A 16 24 34 E1 25 F1 17 18 19 1A 26 27 ; br?.$4??...&'
000001f0h: 28 29 2A 35 36 37 38 39 3A 43 44 45 46 47 48 49 ; ()*56789:CDEFGHI
00000200h: 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 ; JSTUVWXYZcdefghi
00000210h: 6A 73 74 75 76 77 78 79 7A 82 83 84 85 86 87 88 ; jstuvwxyz???口
00000220h: 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 ; ????? 上它
00000230h: A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 ; 夾帶眾斯須號甄藥
00000240h: A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 ; 夾帶眾斯須號甄藥
00000250h: C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E2 ; 齏 均助煙第嶠筌
00000260h: E3 E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 ; 蝟揆碎齏觀鑊論闕
00000270h: FA FF DA 00 0C 03 01 00 02 11 03 11 00 3F 00 ; DA: ??.....?.口
00000280h: BD 8E 88 C1 A2 ED B4 BF 6C C3 1E 36 CF 8C FE FE ; 2諄e曾1?6?
00000290h: 1C 63 F7 C3 19 CA E3 A7 DE 4E 9B 7F 77 4B C7 49 ; 2諄.忻技N?wK
000002a0h: 70 3E 1C 6A A5 AD AE C2 6D 87 E6 69 72 B8 CA F3 ; 2諄.忻技N?wK
000002b0h: 8F 34 F1 CA FF 00 0F F1 27 03 18 8F 3A FB 52 D0 ; 口
000002c0h: 64 83 4B 03 52 D1 D8 AC 2A 1F F7 D0 9D A7 CE 42 ; 口
000002d0h: 73 FB 93 D8 B1 39 CF F1 75 F9 BC CA FE 2F D4 B4 ; 口
000002e0h: 49 BC 0F A8 C5 6D A8 E9 52 5C B7 94 56 38 E6 88 ; 口
000002f0h: 4B D8 AB 5A FE E7 F3 77 66 14 69 D4 58 84 DB 76 ; 口
00000300h: E6 EE FB 33 FF D9 ; 口
```

A. JFIF File Tag

- ✿ *JPEG Standard* 中定義 `0xFF` 為 *Escape code*，由 `0xFF` 之後跟著的資料皆為影像交換的資訊，`0xFF` 後加上一個代碼稱為 *marker*，其他的資訊加在這之後。
- ✿ 如果影像本身 (*MCU* 資料流) 有 `0xFF` 的碼要傳遞，就傳 `0xFF00` 代表。
- ✿ *Restart Interval* 個 *MCUs* 後面要接一個 *RST* *marker*，如果 *MCUs* 的 *bits* 數不是 8 的倍數，則在尾端補 1s 直到 1 byte 為止。

標記號碼	標記名稱	說明
FFD8	<i>SOI</i>	Start Of Image (圖檔的開始標記)
FFD9	<i>EOI</i>	End Of Image (圖檔的結束標記)
FFD0~D7	<i>RSTn</i>	Restart Interval 分界記號

B. Frame Header

<i>Escape</i>	<i>SOF0</i>	<i>Lf</i>		<i>P</i>	<i>Y</i>		<i>X</i>		<i>Nf</i>	<i>Ci</i>	<i>Hi</i>	<i>Vi</i>	<i>Tqi</i>
0xFF	0xC0	0x00	0x11	0x08					0x03				


參數	長度 (bits)	說 明
<i>Lf</i>	16	Frame header 的 bytes 數 (不含 marker)
<i>P</i>	8	Precision，描述一個點的位元數，baseline 為 8
<i>Y</i>	16	影像高度
<i>X</i>	16	影像寬度
<i>Nf</i>	8	色彩分量。單色為 1，彩色為 3。此數亦為後面描述色彩分量特性欄位的組數。
<i>Ci</i>	8	分量的編號，每個分量的編號都不同。
<i>Hi</i>	4	分量的水平取樣比例；其值為 1~4。
<i>Vi</i>	4	分量的垂直取樣比例；其值為 1~4。
<i>Tqi</i>	8	分量使用的量化表編號。解碼時，同一個分量必須使用同一個表；其值為 0 ~ 3。

C. Scan Header


Escape	SOS0	Ls		Ns	Csi	Tdi	Tai	Ss	Se	Ah	Al
0xFF	0xDA	0x00	0x0C	0x03				0x00	0x3F	0	0

參數	長度 (bits)	說明
Ls	16	Scan header 的 bytes 數 (不含 marker)
Ns	8	在此 Scan 中色彩分量數。彩色為 3。是描述分量特性欄位 (下面三欄) 的組數。
Csi	8	分量的編號。Csi 為 Frame Header 所記錄 Ci 中的一個
Tdi	4	記錄某分量 Csi 所使用 DC Huffman Table 的編號。其值為 0~1。
Tai	4	記錄某分量 Csi 所使用 AC Huffman Table 的編號。其值為 0~1。
Ss	8	在 baseline 中設為 0；用於 progressive mode 記錄第一個 DCT 矩陣係數的編號。
Se	8	在 baseline 中設為 63；記錄最後一個 DCT 矩陣係數的編號。
Ah	4	在 baseline 中設為 0。
Al	4	在 baseline 中設為 0。

D. Table/misc

- 

在 baseline 模式中，Table/misc 結構由以下五種單元組成：

 - DQT (Define Quantization Table)：量化表。
 - DHT (Define Huffman Table)：Huffman Table。
 - DRI (Define Restart Interval)：定義每 n 個 MCU 後接上一個 RST 記號。
 - APP (APPLication data)：保留給產生此圖檔的應用程式使用。
 - COM (COMment)：記錄產生此圖檔的應用程式對此圖的注釋。
- 

上述單元在圖檔中可出現一次以上；重複出現時，以新出現的取代原有的。



DQT (Define Quantization Table)

Escape	DQT	Lq		Pq	Tq	Q0 ~ Q63
0xFF	0xDB			0		

參數	長度 (bits)	說明
Lq	16	DQT 單元的 bytes 數 (不含 marker)
Pq	4	定義每個量化表係數的 bits 數量，0 表示 8 bits。當 Frame Header 的 precision 欄位的值為 8 時，Pq 必為 0。
Tq	4	量化表的編號，可能值為 0~3。
Qs	8	以 Zigzag 順序取出量化表的係數。一組有 64 個。

- ✿ 每個 DQT 單元最多可容納 4 個量化表。
- 而一個 DQT 內含 “ $(Lq - 2) / 65$ ” 個量化表。



DHT (Define Huffman Table)

Escape	DHT	Lh		Tc	Th	L1~L16	Vi,j
0xFF	0xC4						

參數	長度 (bits)	說明
Lh	16	DHT 單元的 bytes 數 (不含 marker)
Tc	4	Huffman Table 的分類，0 為 DC，1 為 AC。
Th	4	Huffman Table 的編號，其值為 0~3。
Li	8	記錄長度為 i bits 的 code_word 的個數。
Vi,j	8	對應於長度 i 的 code_word 的第 j 個 symbol 值。

- ✿ 每個 DHT 單元最多可容納 8 個 Huffman Table，但在 baseline 中只有 4 個。
- ✿ 以數學表示， $V_{i,j}$ 的長度為 “Sum (L_i) for $i = 1 \sim 16$ ” 個。

DRI (Define Restart Interval)

Escape	APPn	Lr		R	
0xFF	0xE0~EF	0x00	0x04		

參數	長度 (bits)	說明
Lr	16	DRI 單元的 bytes 數 (不含 marker)
R	16	每個 restart interval 中 MCU 的個數。

- ✿ DRI 單元的長度 (含marker) 只有 6 bytes。
- ✿ R 值若不為零，則表示在此 DRI 單元之後每隔 R 個 MCU 就有一個 RST。
- R 值若為零，則代表在此 DRI 單元之後不會再出現 RST marker。

APP (APPlication Data)

Escape	APPn	Lq		APi
0xFF	0xE0~EF			

參數	長度 (bits)	說明
Lp	16	APP 單元的 bytes 數 (不含 marker)
APi	8	記錄僅供產生圖檔的應用程式使用的資訊。

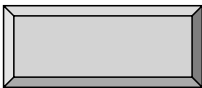





COM (COMment)

Escape	COM	Lc		CMi
0xFF	0xFE			

參數	長度(bits)	說明
Lc	16	COM 單元的 bytes 數 (不含 marker)
CMi	8	記錄僅供產生圖檔的應用程式對此圖檔的註釋。

方塊流程圖 (Block diagram)

圖例說明：

-  演算法。
-  功能區塊，一個區塊表示一種功能。
-  唯讀記憶體。
-  存放 Table 的記憶體。
-  少量的計算參數，大多由 Header 提供。
-  一、二維記憶體。

